

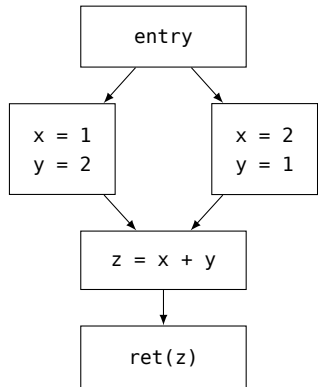
Data-flow analysis

Data-flow analysis

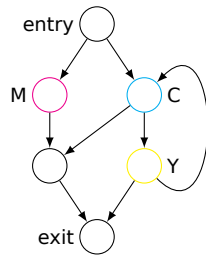
- Static
- Global (whole CFG)
- Control-flow dependent
- Computes run-time properties
- Unified formal model and theory

Applications

- Reaching definitions (use-def links)
- Live-variable analysis
- Constant propagation
- Constant subexpression elimination
- Dead code elimination



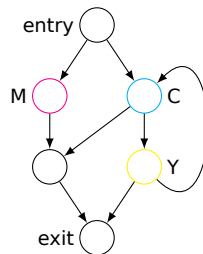
Example



Example

Data-flow framework

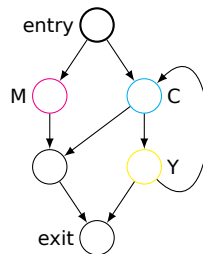
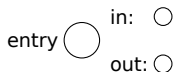
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

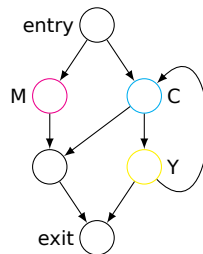
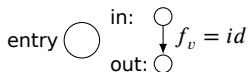
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

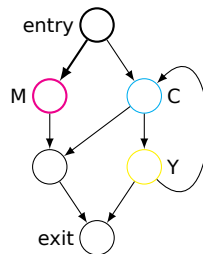
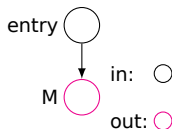
- Flow graph $G = \langle V, E, v_{\text{entry}}, v_{\text{exit}} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $\text{in}_0(v) = \text{out}_0(v) = \top$



Example

Data-flow framework

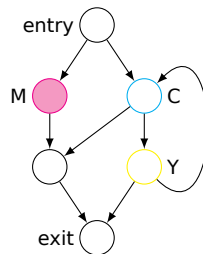
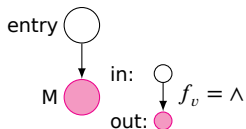
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

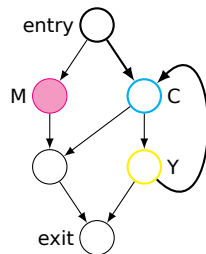
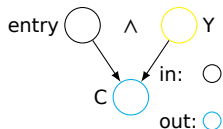
- Flow graph $G = \langle V, E, v_{\text{entry}}, v_{\text{exit}} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

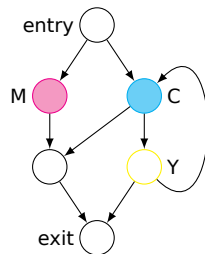
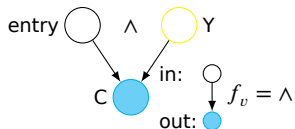
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

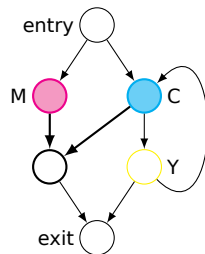
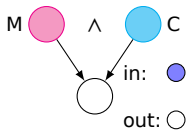
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

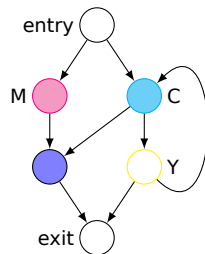
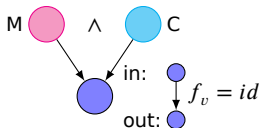
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

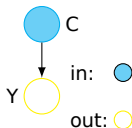
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

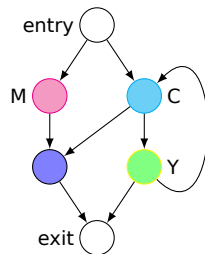
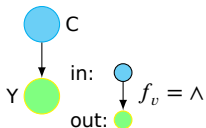
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = T$



Example

Data-flow framework

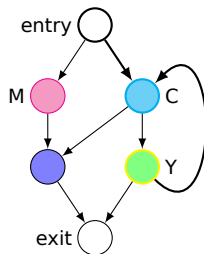
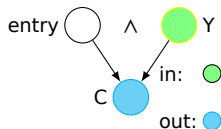
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

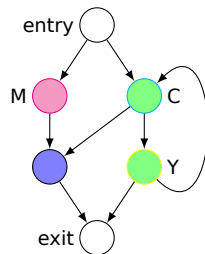
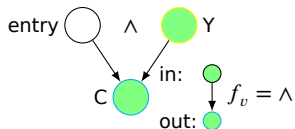
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

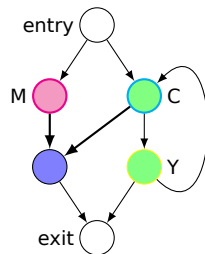
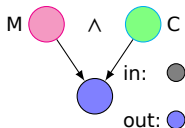
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

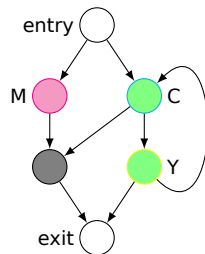
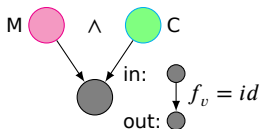
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

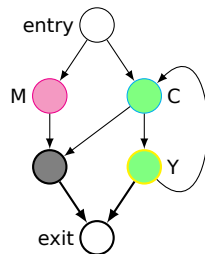
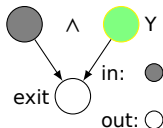
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

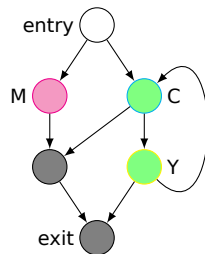
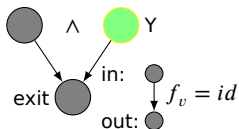
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Example

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Binary operation \wedge (*meet*)

- $x \wedge x = x$ (*idempotency*)
- $x \wedge y = y \wedge x$ (*commutativity*)
- $(x \wedge y) \wedge z = x \wedge (y \wedge z)$ (*associativity*)

Partial order \leq

- $x \leq x$ (*reflexivity*)
- $x \leq y \ \& \ y \leq z \Rightarrow x \leq z$ (*transitivity*)
- $x \leq y \ \& \ y \leq x \Rightarrow x = y$ (*antisymmetry*)

Semilattice $\langle L, \wedge \rangle$ ^{1 2}

- $x \leq y \Leftrightarrow_{def} x \wedge y = x$
- $x < y \Leftrightarrow_{def} x \wedge y = x \ \& \ x \neq y$

¹Do partial order conditions hold for this definition of \leq via \wedge ?

²Is it possible to define semilattice $\langle L, \wedge \rangle$ having only partial order $\langle L, \leq \rangle$?

Binary operation \wedge (meet)

- $x \wedge x = x$ (idempotency)
- $x \wedge y = y \wedge x$ (commutativity)
- $(x \wedge y) \wedge z = x \wedge (y \wedge z)$ (associativity)

Partial order \leq

- $x \leq x$ (reflexivity)
- $x \leq y \ \& \ y \leq z \Rightarrow x \leq z$ (transitivity)
- $x \leq y \ \& \ y \leq x \Rightarrow x = y$ (antisymmetry)

Semilattice $\langle L, \wedge \rangle$ ^{1 2}

- $x \leq y \Leftrightarrow_{def} x \wedge y = x$
- $x < y \Leftrightarrow_{def} x \wedge y = x \ \& \ x \neq y$

Upper bound

$$\exists \perp \in L : \forall x \in L : \perp \wedge x = \perp \ (\perp \leq x)$$

Lower bound

$$\exists \top \in L : \forall x \in L : \top \wedge x = x \ (x \leq \top)$$

Semilattice height

$$H_L = \max\{|x_1 > x_2 > \dots \in L|\}$$

Descending chain condition

$$\forall x_1 > x_2 > \dots \in L : \exists k : \nexists y \in L : x_k > y$$

Semilattice product

$$\begin{aligned} \langle A, \wedge_A \rangle \times \langle B, \wedge_B \rangle &= \langle A \times B, \wedge \rangle, \\ (a, b) \wedge (a', b') &= (a \wedge_A a', b \wedge_B b') \end{aligned}$$

¹Do partial order conditions hold for this definition of \leq via \wedge ?

²Is it possible to define semilattice $\langle L, \wedge \rangle$ having only partial order $\langle L, \leq \rangle$?

Power set of S

$$L = 2^S, \wedge = \cup \text{ или } \cap$$

Power set of S

$$L = 2^S, \wedge = \cup \text{ или } \cap$$

$$\emptyset = \top$$

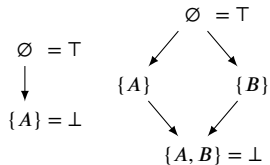


$$\{A\} = \perp$$

Examples

Power set of S

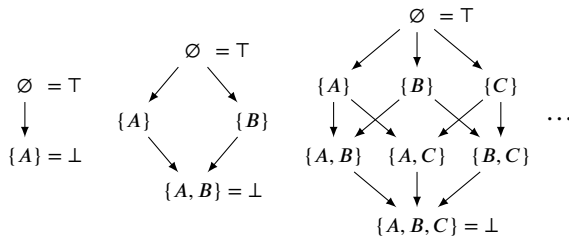
$$L = 2^S, \wedge = \cup \text{ или } \cap$$



Examples

Power set of S

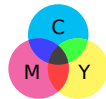
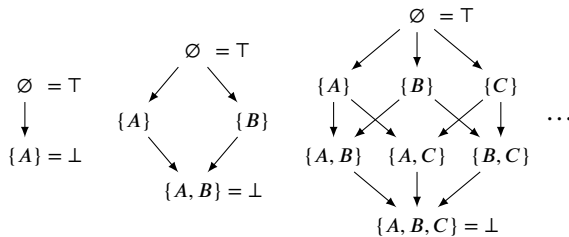
$$L = 2^S, \wedge = \cup \text{ или } \cap$$



Examples

Power set of S

$$L = 2^S, \wedge = \cup \text{ или } \cap$$



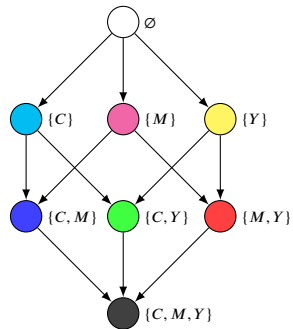
Examples

Power set of S

$$L = 2^S, \wedge = \cup \text{ или } \cap$$

CMYK

$$L = 2^{\{C,M,Y\}}, \wedge = \cup$$



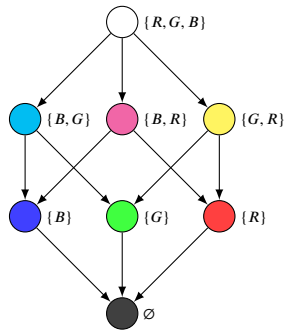
Examples

Power set of \mathcal{S}

$$L = 2^{\mathcal{S}}, \wedge = \cup \text{ или } \cap$$

RGB

$$L = 2^{\{R,G,B\}}, \wedge = \cap$$



Examples

Power set of S

$$L = 2^S, \wedge = \cap \text{ или } \cup$$

Natural numbers

$$L = \mathbb{N}_0 \cup \{\top\}, x \wedge y = \min(x, y)$$

\top
 \downarrow
 \dots
 \downarrow
 2
 \downarrow
 1
 \downarrow
 0

Examples

Power set of S

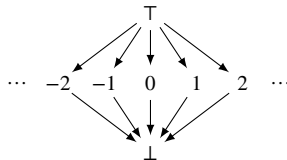
$$L = 2^S, \wedge = \cap \text{ или } \cup$$

Natural numbers

$$L = \mathbb{N}_0 \cup \{T\}, x \wedge y = \min(x, y)$$

Integer constants

$$L = \mathbb{Z} \cup \{T, \perp\}, \perp < \mathbb{Z} < T$$



Examples

Power set of S

$$L = 2^S, \wedge = \cap \text{ или } \cup$$

Natural numbers

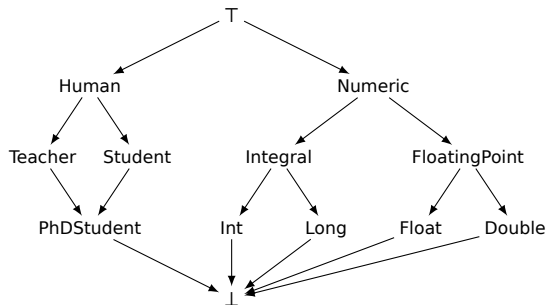
$$L = \mathbb{N}_0 \cup \{T\}, x \wedge y = \min(x, y)$$

Integer constants

$$L = \mathbb{Z} \cup \{T, \perp\}, \perp < \mathbb{Z} < T$$

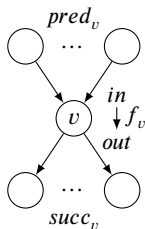
Program type hierarchy

$$L = \text{Types}, x \leq y \Leftrightarrow x <: y$$



Data-flow framework

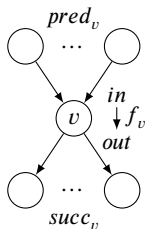
- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Data-flow framework

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$



Data-flow equations

$D = \downarrow$	$D = \uparrow$
$in_0(v) = out_0(v) = \top$	$in_0(v) = out_0(v) = \top$
$in_i(v) = \bigwedge_{x \in pred_v} out_i(x)$	$out_i(v) = \bigwedge_{x \in succ_v} in_i(x)$
$out_i(v) = f_v(in_i(v))$	$in_i(v) = f_v(out_i(v))$

Maximum Fixed Point (MFP)

Maximum solution among all solutions S

$$out_S(v) \leq out_{MFP}(v) \quad | \quad in_S(v) \leq in_{MFP}(v)$$

Convergence conditions

- Monotonicity of transfer functions f_v
- Meet-semilattice $\langle L, \wedge \rangle$ with descending chains condition

Data-flow framework

Data-flow framework

- Flow graph $G = \langle V, E, v_{entry}, v_{exit} \rangle$
- Direction of analysis $D \in \{\downarrow, \uparrow\}$
- Meet-semilattice $\langle L, \wedge \rangle$ with upper bound
- Transfer functions $f_{v \in V} : L \rightarrow L$
- Boundary condition $in_0(v) = out_0(v) = \top$

Transfer functions

Monotone function f on $\langle L, \leq \rangle$

$$x \leq y \Rightarrow f(x) \leq f(y)$$

Monotone function f on $\langle L, \wedge \rangle$ ³

$$f(x \wedge y) \leq f(x) \wedge f(y)$$

Distributive function f on $\langle L, \wedge \rangle$

$$f(x \wedge y) = f(x) \wedge f(y)$$

Data-flow equations

$D = \downarrow$	$D = \uparrow$
$in_0(v) = out_0(v) = \top$	$in_0(v) = out_0(v) = \top$
$in_i(v) = \bigwedge_{x \in pred_v} out_i(x)$	$out_i(v) = \bigwedge_{x \in succ_v} in_i(x)$
$out_i(v) = f_v(in_i(v))$	$in_i(v) = f_v(out_i(v))$

Maximum Fixed Point (MFP)

Maximum solution among all solutions S

$$out_S(v) \leq out_{MFP}(v) \quad | \quad in_S(v) \leq in_{MFP}(v)$$

Convergence conditions

- Monotonicity of transfer functions f_v
- Meet-semilattice $\langle L, \wedge \rangle$ with descending chains condition

³Prove the equivalence of given monotone function definitions on $\langle L, \leq \rangle$ and on $\langle L, \wedge \rangle$.

Examples of divergent analysis

Monotonicity of transfer functions

$$L = \{T, F\}, F \leq T$$

$$f_{\text{entry}} = f_{\text{exit}} = \text{id}$$

$$f_{\text{loop}}(x) = \neg x$$

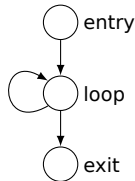
Descending chain condition ⁴

$$L = \mathbb{R}_0^+ \cup \{T\}, \wedge = \text{min}$$

$$f_{\text{entry}}(x) = 1$$

$$f_{\text{loop}}(x) = x/2$$

$$f_{\text{exit}} = \text{id}$$



⁴Can there be a semilattice with unlimited height which satisfies descending chain condition?

Meet Over Paths (MOP) ⁵

Precise solution over all paths $v_{entry} \rightarrow \dots \rightarrow v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \rightarrow \dots \rightarrow v} f_v(\dots (f_{v_{entry}}(\top)) \dots)$$

MFP safety

$$out_{MFP}(v) \leq out_{MOP}(v)$$

⁵Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

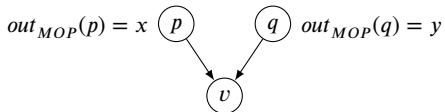
Meet Over Paths (MOP) ⁵

Precise solution over all paths $v_{entry} \rightarrow \dots \rightarrow v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \rightarrow \dots \rightarrow v} f_v(\dots (f_{v_{entry}}(\top)) \dots)$$

MFP safety

$$out_{MFP}(v) \leq out_{MOP}(v)$$



⁵Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

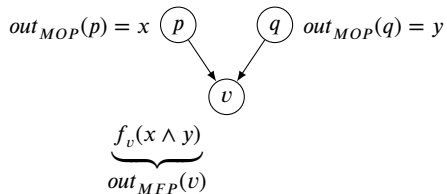
Meet Over Paths (MOP) ⁵

Precise solution over all paths $v_{entry} \rightarrow \dots \rightarrow v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \rightarrow \dots \rightarrow v} f_v(\dots (f_{v_{entry}}(\top)) \dots)$$

MFP safety

$$out_{MFP}(v) \leq out_{MOP}(v)$$



⁵Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

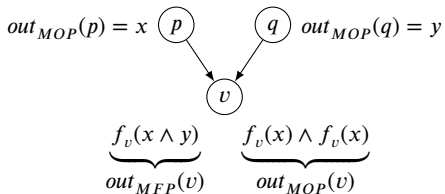
Meet Over Paths (MOP) ⁵

Precise solution over all paths $v_{entry} \rightarrow \dots \rightarrow v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \rightarrow \dots \rightarrow v} f_v(\dots (f_{v_{entry}}(\top)) \dots)$$

MFP safety

$$out_{MFP}(v) \leq out_{MOP}(v)$$



⁵Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

Meet Over Paths (MOP) ⁵

Precise solution over all paths $v_{entry} \rightarrow \dots \rightarrow v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \rightarrow \dots \rightarrow v} f_v(\dots (f_{v_{entry}}(\top)) \dots)$$

MFP safety

$$out_{MFP}(v) \leq out_{MOP}(v)$$

$$\begin{array}{c} out_{MOP}(p) = x \quad \textcircled{p} \quad \textcircled{q} \quad out_{MOP}(q) = y \\ \quad \searrow \quad \swarrow \\ \quad \textcircled{v} \\ \underbrace{f_v(x \wedge y)}_{out_{MFP}(v)} \leq \underbrace{f_v(x) \wedge f_v(y)}_{out_{MOP}(v)} \end{array}$$

⁵Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

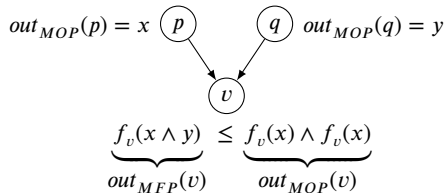
Meet Over Paths (MOP) ⁵

Precise solution over all paths $v_{entry} \rightarrow \dots \rightarrow v$

$$out_{MOP}(v) = \bigwedge_{v_{entry} \rightarrow \dots \rightarrow v} f_v(\dots (f_{v_{entry}}(\top)) \dots)$$

MFP safety ⁶

$$out_{MFP}(v) \leq out_{MOP}(v)$$



⁵Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

⁶If transfer functions are distributive, then MFP is always precise — $out_{MFP}(v) = out_{MOP}(v)$.

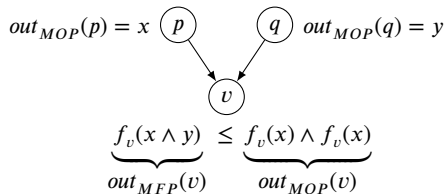
Meet Over Paths (MOP) ⁵

Precise solution over all paths $v_{entry} \rightarrow \dots \rightarrow v$

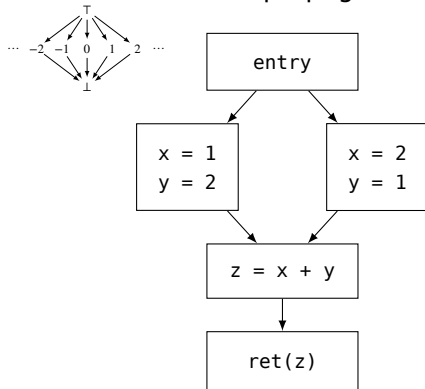
$$out_{MOP}(v) = \bigwedge_{v_{entry} \rightarrow \dots \rightarrow v} f_v(\dots (f_{v_{entry}}(\top)) \dots)$$

MFP safety ⁶

$$out_{MFP}(v) \leq out_{MOP}(v)$$



Constant propagation



⁵Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

⁶If transfer functions are distributive, then MFP is always precise — $out_{MFP}(v) = out_{MOP}(v)$.

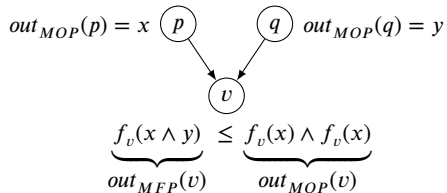
Meet Over Paths (MOP) ⁵

Precise solution over all paths $v_{entry} \rightarrow \dots \rightarrow v$

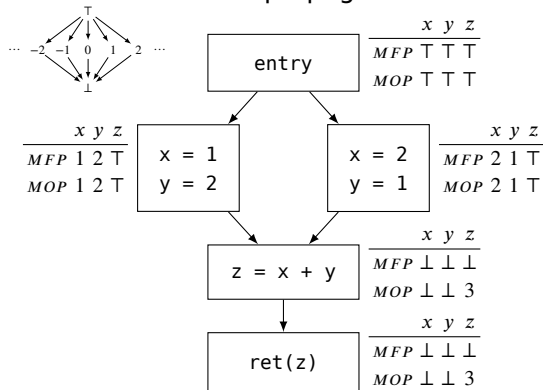
$$out_{MOP}(v) = \bigwedge_{v_{entry} \rightarrow \dots \rightarrow v} f_v(\dots (f_{v_{entry}}(\top)) \dots)$$

MFP safety ⁶

$$out_{MFP}(v) \leq out_{MOP}(v)$$



Constant propagation



⁵Here we only consider forward-flow analysis $D = \downarrow$, the case of backwards-flow analysis $D = \uparrow$ is the same.

⁶If transfer functions are distributive, then MFP is always precise — $out_{MFP}(v) = out_{MOP}(v)$.

Control-flow graph

- $CFG = \langle B, E, entry, exit \rangle$
- Every block $b \in B$ contains exactly one operation
- V — set of variables in a program
- $def_v \subseteq B$ — set of *assignments* into variable $v \in V$
(e.g. $v = 3$)
- $use_v \subseteq B$ — set of *uses* of variable $v \in V$
(e.g. $x = y + v$)

Control-flow graph

- $CFG = \langle B, E, entry, exit \rangle$
- Every block $b \in B$ contains exactly one operation
- V — set of variables in a program
- $def_v \subseteq B$ — set of *assignments* into variable $v \in V$
(e.g. $v = 3$)
- $use_v \subseteq B$ — set of *uses* of variable $v \in V$
(e.g. $x = y + v$)

Gen-Kill formalism

- $L = 2^S, \wedge = \cup$ or \cap
- $f_b(x) = gen_b \cup (x \setminus kill_b)$
- gen_b — properties *generated* by block b
- $kill_b$ — properties *killed* by block b

Control-flow graph

- $CFG = \langle B, E, entry, exit \rangle$
- Every block $b \in B$ contains exactly one operation
- V — set of variables in a program
- $def_v \subseteq B$ — set of *assignments* into variable $v \in V$ (e.g. $v = 3$)
- $use_v \subseteq B$ — set of *uses* of variable $v \in V$ (e.g. $x = y + v$)

Gen-Kill formalism

- $L = 2^S, \wedge = \cup$ or \cap
- $f_b(x) = gen_b \cup (x \setminus kill_b)$
- gen_b — properties *generated* by block b
- $kill_b$ — properties *killed* by block b

Result

- $\langle L, \wedge \rangle$ — finite semilattice
- f_b — distributive functions ⁷
- Analysis *always* converges to precise solution

⁷Prove distributivity of f_b in gen-kill form.

Control-flow graph

- $CFG = \langle B, E, entry, exit \rangle$
- Every block $b \in B$ contains exactly one operation
- V — set of variables in a program
- $def_v \subseteq B$ — set of *assignments* into variable $v \in V$ (e.g. $v = 3$)
- $use_v \subseteq B$ — set of *uses* of variable $v \in V$ (e.g. $x = y + v$)

Reaching definitions

$$L = 2^B, \wedge = \cup, D = \downarrow$$

b	$\in def_v$	$\notin def_v$
gen_b	$\{b\}$	\emptyset
$kill_b$	def_v	\emptyset

Live-variable analysis

$$L = 2^V, \wedge = \cup, D = \uparrow$$

gen_b	$\{v \mid b \in use_v\}$
$kill_b$	$\{v \mid b \in def_v\}$

Gen-Kill formalism

- $L = 2^S, \wedge = \cup \text{ or } \cap$
- $f_b(x) = gen_b \cup (x \setminus kill_b)$
- gen_b — properties *generated* by block b
- $kill_b$ — properties *killed* by block b

Result

- $\langle L, \wedge \rangle$ — finite semilattice
- f_b — distributive functions ⁷
- Analysis *always* converges to precise solution

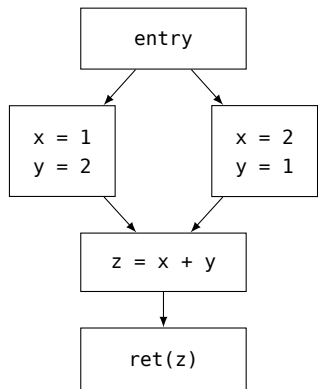
⁷Prove distributivity of f_b in gen-kill form.

Benefits

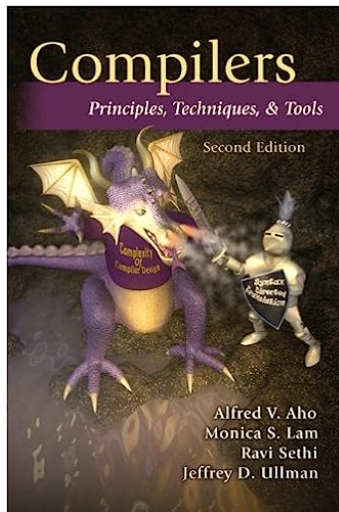
- Global static analysis
- Universal theoretical model
- Straightforward implementation
- gen-kill formalism guarantees convergence and precision

Drawbacks

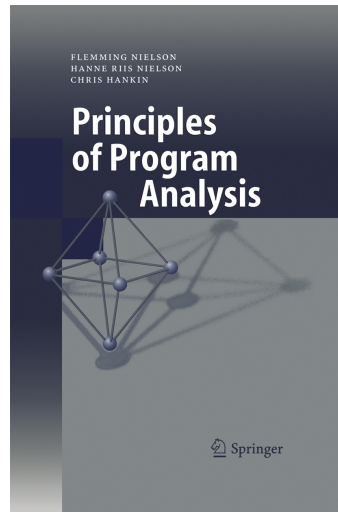
- Result gets invalidated after optimizations
- Analyses do not compose quite efficiently
- Convergence and precision are not guaranteed in general



A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman.
Compilers: Principles, Techniques, and Tools, 1986
Introduction to Data-Flow Analysis



N. Flemming, H. R. Nielson, and C. Hankin.
Principles of program analysis. 2015
Data Flow Analysis



Thank you for attention